CHI Systems, Inc.
100 Burns Place
Goleta, CA 93117
(805) 964-8868

August 1982

CHI-5 SIMULATOR
REFERENCE MANUAL

Quarterly Technical Report

10 May 1982 - 10 August 1982

PRINCIPAL INVESTIGATOR: Dr. Glen J. Culler

PROJECT SCIENTIST: Thomas W. Fuller

This research was supported by the Defense Advanced Research Projects Agency under ARPA Order No. 3625 Contract No. MDA 903-82-C-0136.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER MDA 903-82-C-0136-Q2	2. GOVT ACCESSION NO. AD-A119139	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) CHI-5 Simulator Reference Manual		5. TYPE OF REPORT & PERIOD COVERED Quarterly Technical Report 10 May 1982 - 10 Aug 1982 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) T. W. Fuller		8. CONTRACT OR GRANT NUMBER(s) MDA 903-82-C-0136
9. PERFORMING ORGANIZATION NAME AND ADDRESS CHI Systems, Inc. 100 Burns Place Goleta, CA 93117		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS ARPA Order No. 3625
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Supply Service - Washington Room 1D245, The Pentagon Washington, D.C. 20110 (T. Bushell)		12. REPORT DATE 10 Aug 1982 13. NUMBER OF PAGES 30
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) DCASMA Van Nuys (S0512A) 6230 Van Nuys Blvd. Van Nuys, CA 91408		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Simulator, Array Processor

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This document is a reference manual for the CHI-5 Simulator computer program. It includes directions for installing and running the simulator and a description of the underlying model of the CHI-5.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

CHI-5 SIMULATOR REFERENCE MANUAL

T. W. FULLER
CHI-SYSTEMS, INC.
100 EDWARD BURNS PLACE
GOLETA, CALIFORNIA 93117

AUGUST 27, 1982

PREFACE

This document describes installation and use of the CHI-5 Simulator. The user should be familiar with the architecture of the CHI-5 Array Processor, and may benefit from the CHI-5 Microprogramming Reference Manual.

Chapter 1 of this manual gives an overview of the simulator. Chapter 2 describes the general procedure for using the simulator. Chapter 3 describes the model of the CHI-5 which is maintained by the simulator. Chapter 4 describes the purpose and format for each file used by the simulator. Chapter 5 describes the syntax and function of each command available to the user. Appendix A describes the procedure for installing the simulator in a system.

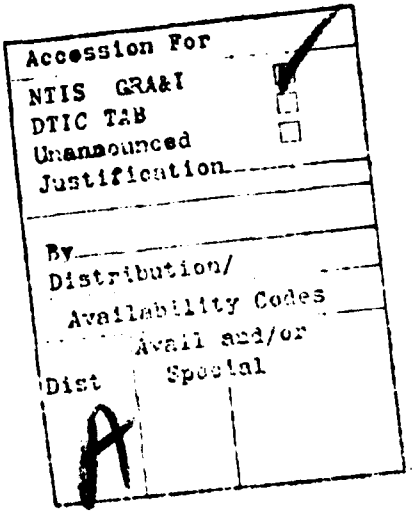


TABLE OF CONTENTS

1. INTRODUCTION

The CHI-5 Simulator is an interactive program, written in standard FORTRAN-77, which simulates the operation of a CHI-5 Array Processor. It may be used to verify the correctness of a microprogram designed to run on the CHI-5.

The simulator maintains a functional model of the CHI-5. This model is composed of variables, known as "state variables", which hold values corresponding to values held by elements in a real CHI-5. A set of commands is available to the user with which the values of state variables may be manipulated. Values may be changed directly, or through the simulated execution of CHI-5 instructions.

A system of files and the commands to control them are provided so values in the model may be more efficiently loaded or stored.

2. RUNNING THE SIMULATOR

The simulator may be run either interactively from a terminal, or as a batch job. It is started by the operating system like any other FORTRAN-77 program. When it first starts, a message is sent to the System Output Device indicating the version and date of the simulator which is being run, and any configuration differences (e.g. non-standard memory sizes) in effect. Then the simulator goes into an interactive mode in which the prompt "CHI5*" is sent to the System Output Device, and the simulator reads a command line from the System Input Device.

One or more fixed length commands may be present in a command line. When all the commands on the line have been executed by the simulator the "CHI5*" prompt is sent to the System Output Device again, and another command line is read. When the command "EXIT" is found the simulation is terminated.

Error messages are written to the System Output File, as are command responses and displayed simulation results. The error messages may indicate command syntax errors, or inconsistencies in the simulation, and are designed to be self explanatory.

3. CHI-5 MODEL

The simulator maintains a model of the CHI-5 in a set of variables which hold values corresponding to the registers, busses, and memory elements of the CHI-5. These variables are called "state variables", because their values tell the user the state of the CHI-5 model.

State variable names are derived from the names of the elements they represent, and are used during a simulation to refer to an element's value. A list of the state variable names and their corresponding elements is given in table 3.1. A drawing of the CHI-5 architecture using state variable names appears in figure 3.1. Table 3.2 shows the location of each field within the instruction word.

The state of the model may be examined by the user between the simulated execution of CHI-5 instructions. At the time of examination the current instruction will have been placed in the instruction register, the XB, YB, RBUS, DABUS buses and all registers will contain values as specified by the previous instruction, and the VAL bus will contain a value as specified by the current instruction.

The user may not examine the instruction register, but may look at the current instruction as it sits in program memory at the address specified by PSA or PHA. The user may modify the contents of memory elements or registers, but may not modify the contents of buses, because the values will be discarded as the buses assume new values during the simulated execution of an instruction.

The 10 S bits are set to 1 only by the user, and cleared to zero only by the simulated execution of an instruction with an operation to do so.

DMA transfers are simulated by the LOAD and STORE commands for D-memory without effecting the DA registers, S bits or interrupt signals.

"Analog" data may be transferred between the analog interface unit (device 8) and files acting as the "outside world". These files should contain a list of hexadecimal integers.

When the LOAD ANALOG or STORE ANALOG command is given, the Analog Input or Analog Output File, resp., is named and opened, and is available to send or receive, resp., data which is "time-released" through the Analog Interface.

The simulator maintains a clock for analog I/O which 'ticks' once every 500 instructions. Each time it ticks a value may

be transferred from the Analog Input File to the A/D FIFO, or from the D/A FIFO to the Analog Output File, if either of those files is open.

The serial line interface (device 9) is simulated with a 16 bit register named "SLI", which may be set and examined by the user as well as the simulator. The user may also set S09 to cause interrupt. For devices 0-7, and 10-15: 1) XB=IO causes XB to be set to an undefined value, and 2) XB->IO causes NOOP.

The simulator performs rounding during a multiplication in FR mode by adding a 1 to bit 14 of the multiplier result. Bits 15-0 of the result are placed in MR, and bits 30-15 are placed in ML.

The high order bit of PSA selects RAM (0) or ROM (1). When ROM is selected the address is taken from PHA.

When the Interrupt Enable bit is set and at least one S bit is set then the next EXEC MACRO operation will cause 40 (HEX) to be placed in BCNTR.

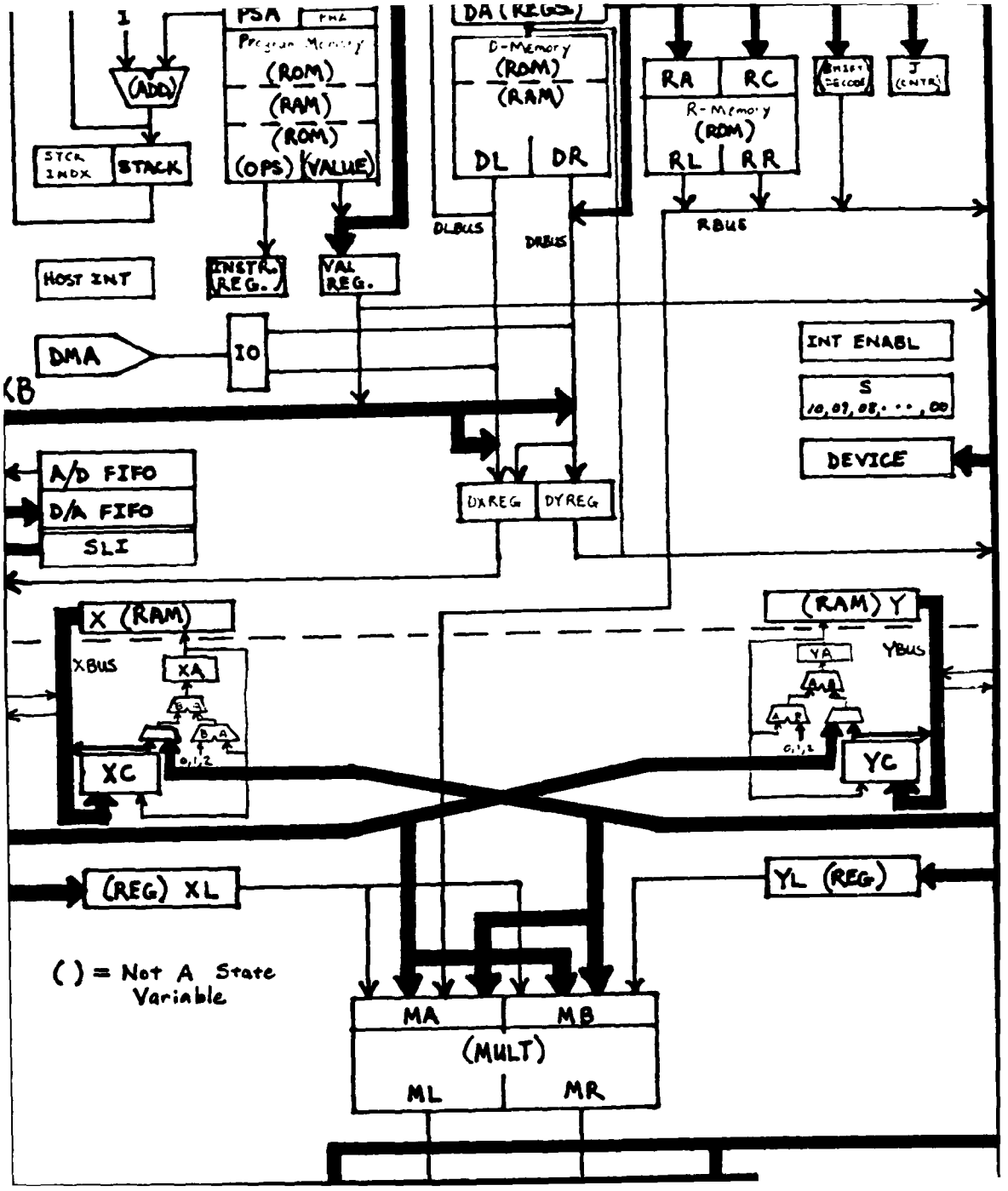
The RESET command forces 100 (HEX) onto the VAL bus.

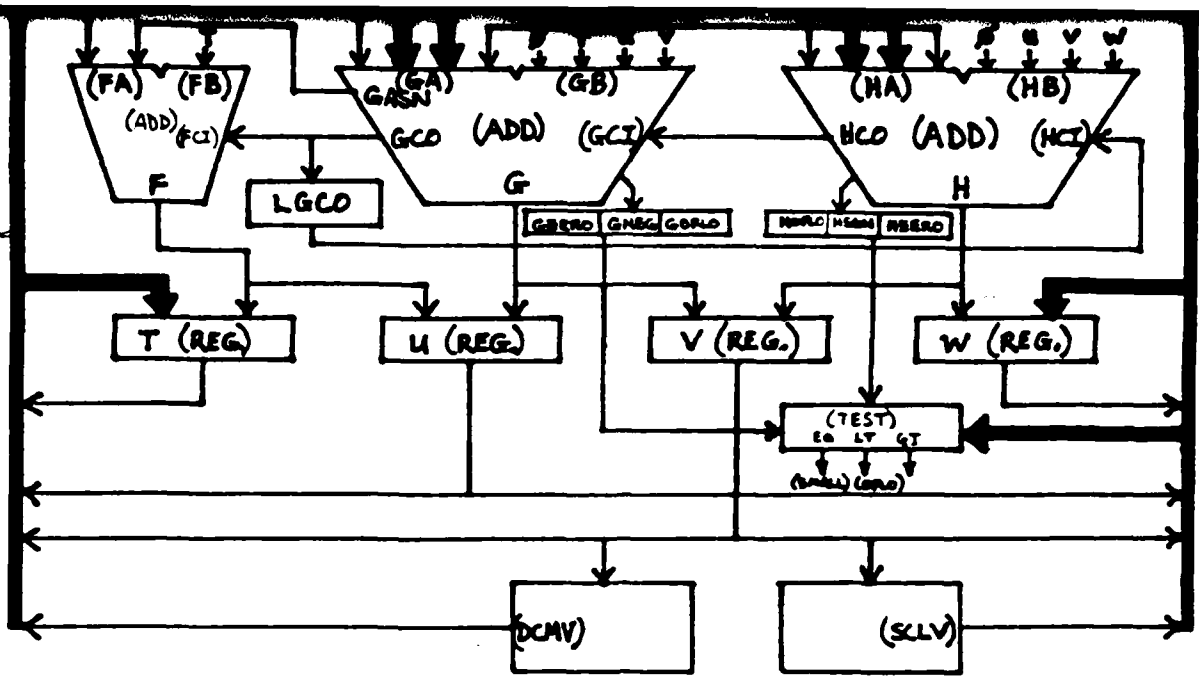
NAME	WIDTH	CONTENTS
ADFIFO	16 BITS	A/D FIFO
BCNTR	8 BITS	PROGRAM ADDRESS COUNTER
D	16 BITS	D-MEMORY
DA	16 BITS	D-MEMORY ADDRESS FILE
DABUS	16 BITS	D-MEMORY ADDRESS BUS
DAFIFO	16 BITS	D/A FIFO
DAINDEX	5 BITS	D-MEMORY ADDRESS FILE INDEX
DEVICE	4 BITS	DEVICE REGISTER
DLBUS	16 BITS	D-MEMORY LEFT BUS
DRBUS	16 BITS	D-MEMORY RIGHT BUS
DXREG	16 BITS	DX REGISTER
DYREG	16 BITS	DY REGISTER
EQ	1 BIT	TEST RESULT (0=(NE),1=(EQ))
F	16 BITS	F-ADDER RESULT
G	16 BITS	G-ADDER RESULT
GASN	1 BIT	G-ADDER A-INPUT SIGN
GCO	1 BIT	G-ADDER CARRY OUT
GNEG	1 BIT	G-ADDER NEGATIVE STATUS
GOFLO	1 BIT	G-ADDER OVERFLOW STATUS
GZERO	1 BIT	G-ADDER ZERO STATUS
H	16 BITS	H-ADDER RESULT
HCO	1 BIT	H-ADDER CARRY OUT
HOFLO	1 BIT	H-ADDER OVERFLOW STATUS
HOSTINT	1 BIT	HOST INTERRUPT BIT
HSGN	1 BIT	H-ADDER BIT 17
HZERO	1 BIT	H-ADDER ZERO STATUS
INTENABL	1 BIT	INTERRUPT ENABLE BIT
J	16 BITS	J COUNTER
LGCO	1 BIT	LAST GCO-REGISTER
LT	1 BIT	TEST RESULT (0=(GE),1=(LT))
MA	17 BITS	MULTIPLIER A-INPUT LATCH WITH SIGN CONTROL (MOST SIGNIFICANT BIT: 0 = POSITIVE, 1 = 2'S COMPL.)
MB	17 BITS	MULTIPLIER B-INPUT LATCH WITH SIGN CONTROL (MOST SIGNIFICANT BIT: 0 = POSITIVE, 1 = 2'S COMPL.)
ML	16 BITS	MULTIPLIER RESULT (LEFT) FOR INPUT BEFORE LAST
MLMID	16 BITS	MULTIPLIER RESULT (LEFT) FOR LAST INPUT
MR	16 BITS	MULTIPLIER RESULT (RIGHT) WITH SIGN OF MPL FOR INPUT BEFORE LAST
MRMID	16 BITS	MULTIPLIER RESULT (RIGHT) WITH SIGN OF MPL FOR LAST INPUT

TABLE 3.1-a STATE VARIABLE NAMES AND CORRESPONDING ELEMENTS

NAME	WIDTH	CONTENTS
PHA	11 BITS	PROGRAM ROM ADDRESS BUS
P	80 BITS	PROGRAM MEMORY
PSA	12 BITS	PROGRAM ADDRESS REGISTER
R	16 BITS	R-MEMORY
RBUS	16 BITS	R-BUS
S00	1 BIT	STATUS BIT
S01	1 BIT	STATUS BIT
S02	1 BIT	STATUS BIT
S03	1 BIT	STATUS BIT
S04	1 BIT	STATUS BIT
S05	1 BIT	STATUS BIT
S06	1 BIT	STATUS BIT
S07	1 BIT	STATUS BIT
S08	1 BIT	STATUS BIT
S09	1 BIT	STATUS BIT
SLI	16 BITS	SERIAL LINE INTERFACE
STACK	12 BITS	SUBROUTINE RETURN ADDRESS STACK
STCKINDX	4 BITS	STACK INDEX
T	16 BITS	T-REGISTER
U	16 BITS	U-REGISTER
V	16 BITS	V-REGISTER
VALBUS	16 BITS	VALUE BUS
VALREG	16 BITS	VALUE REGISTER
W	16 BITS	W-REGISTER
XA	16 BITS	XA-REGISTER
XB	16 BITS	X-BUS
XBUS	16 BITS	X-XC BUS
XC	16 BITS	XC-REGISTER
XL	16 BITS	X-LATCH REGISTER
X	16 BITS	X MEMORY
YA	16 BITS	YA-REGISTER
YB	16 BITS	Y-BUS
YBUS	16 BITS	Y-YC BUS
YC	16 BITS	YC-REGISTER
YL	16 BITS	Y-LATCH REGISTER
Y	16 BITS	Y MEMORY

TABLE 3.1-b STATE VARIABLE NAMES AND CORRESPONDING ELEMENTS


1
(ADD)
PSA
Program Memory
(ROM)
(RAM)
(ROM)
(OPS)
(VALUE)
STCK INDX
STACK
DA (REGS)
D-Memory
(ROM)
(RAM)
DL
DR
RA
RC
R-Memory
(ROM)
RL
RR
J (CNTR)
DLBUS
DRBUS
RBUS
HOST INT
INSTR. REG.
VAL REG.
DMA
IO
INT ENABL
S
DEVICE
A/D FIFO
D/A FIFO
SLI
DXREG
DYREG
X (RAM)
XBUS
XA
XC
(RAM) Y
YA
YBUS
YC
(REG) XL
YL (REG)
( ) = Not A State Variable
MA
MB
(MULT)
ML
MR


(FA)
(FB)
(ADD)
(FCI)
F
LGCO
(GA)
GASN
(GB)
GCO
(ADD)
(GCI)
G
GZERO
GNEG
GOFLO
(HA)
(HB)
HCO
(ADD)
(HCI)
H
HZERO
T (REG.)
U (REG.)
V (REG.)
W (REG.)
(TEST)
EQ LT GT
(DCMV)
(SCLV)

FIELD	POSITION IN WORD (BITS ON)		
VALUE	0000FFFF	00000000	00000000
PSA	00000000	F0000000	00000000
TEST	00000000	0C000000	00000000
R	00000000	02000000	00000000
RA	00000000	01000000	00000000
M	00000000	00E00000	00000000
MULT	00000000	001F0000	00000000
D	00000000	000F0000	00000000
HCI	00000000	00040000	00000000
RLD	00000000	00030000	00000000
IO	00000000	00030000	00000000
F	00000000	0000E000	00000000
FG	00000000	00001000	00000000
G	00000000	00000F00	00000000
GH	00000000	00000080	00000000
HA	00000000	00000060	00000000
H	00000000	0000001C	00000000
XL	00000000	00000002	00000000
YL	00000000	00000001	00000000
XB	00000000	00000000	E0000000
GB	00000000	00000000	18000000
TU	00000000	00000000	07000000
XA	00000000	00000000	00F00000
X	00000000	00000000	000F0000
YB	00000000	00000000	0000E000
HB	00000000	00000000	00001800
VW	00000000	00000000	00000700
YA	00000000	00000000	000000F0
Y	00000000	00000000	0000000F

TABLE 3.2-a OPERATION FIELDS WITHIN INSTRUCTION FROM LEFT TO RIGHT.

FIELD	POSITION IN WORD (BITS ON)		
D	00000000	000F0000	00000000
F	00000000	0000E000	00000000
FG	00000000	00001000	00000000
G	00000000	00000F00	00000000
GB	00000000	00000000	18000000
GH	00000000	00000080	00000000
H	00000000	0000001C	00000000
HA	00000000	00000060	00000000
HB	00000000	00000000	00001800
HCI	00000000	00040000	00000000
IO	00000000	00030000	00000000
M	00000000	00E00000	00000000
MULT	00000000	001F0000	00000000
PSA	00000000	F0000000	00000000
R	00000000	02000000	00000000
RA	00000000	01000000	00000000
RLD	00000000	00030000	00000000
TEST	00000000	0C000000	00000000
TU	00000000	00000000	07000000
VALUE	0000FFFF	00000000	00000000
VW	00000000	00000000	00000700
X	00000000	00000000	000F0000
XA	00000000	00000000	00F00000
XB	00000000	00000000	E0000000
XL	00000000	00000002	00000000
Y	00000000	00000000	0000000F
YA	00000000	00000000	000000F0
YB	00000000	00000000	0000E000
YL	00000000	00000001	00000000

TABLE 3.2-b. OPERATION FIELDS WITHIN INSTRUCTION (ALPHABETICAL).

4. I/O FILES

The CHI-5 Simulator communicates with up to fifteen different files. These files are divided into 3 classes: Control, State Variable, and Logging. The files are assigned to logical units through parameters within the simulator program (see APPENDIX A. INSTALLATION), and are opened and closed automatically, so the user need only be concerned with file names.

The control files are always available to the simulator. The names of files to be used as State Variable files, and Logging files are given interactively by the user, and are the same file names expected by the operating system. The assignment of these names is described in chapter 5 under the appropriate command.

All files are text, and their maximum record lengths have been set to 80 characters unless changed during installation (see APPENDIX A. INSTALLATION).

4.1 CONTROL FILES

There are two types of control files: 1) the System Input File, and 2) the System Output File. The System Input File provides the commands which control the simulation process. The System Output File accepts the output from the simulator indicating the results of a command.

4.1.1 SYSTEM INPUT FILE

The System Input File is expected to be available on the System Input Device. When the CHI-5 Simulator is run as a batch job, then the System Input File is indeed a file, but when run interactively becomes the sequence of characters entered on the terminal during the run. The text in this file is interpreted as commands to the CHI-5 Simulator. Further explanation may be found in Chapter 5. COMMANDS.

4.1.2 SYSTEM OUTPUT FILE

The System Output File is expected to be available on the System Output Device. When the CHI-5 Simulator is run the file is sent responses to commands including error messages and state variable values for variables whose display flags are "true". When the simulator is run as a batch job the System Output File is indeed a file, but when run interactively becomes the sequence of characters displayed on the terminal.

4.2 STATE VARIABLE FILES

There are a large number of memory elements in the CHI-5 Array Processor, and to set each of their values manually during a simulation would be time consuming. The loading (storing) of these memory values may be done from (to) State Variable files of the appropriate type.

There are 3 formats of State Variable Files: 1) Data, 2) Program, and 3) State. Their contents are summarized in table 4.1. The names of the most recent input and output files is remembered for each memory type and for the state variables. These names may be displayed by the SHOW FILES command as a reminder.

File Format	Contents
Data File	D-, R-, X-, or Y-Memory values
Program File	P-Memory values
State File	State Variable values

Table 4.1 State Variable File Contents.

4.2.1 DATA FILES

A Data File contains values corresponding to those contained in the simulator's D-, R-, X-, and Y-memories. The values in the file may have been generated by an editor or some other program, or the file may have received data from the simulator as an output file. The contents of the simulator's D-, R-, X-, or Y-memory may be loaded from, or stored to a Data File on a command from the user.

The file is composed of a sequence of records. Each record contains one or more hexadecimal values separated by at least one non-alphanumeric character. Blank records are ignored. The file is terminated by an end-of-file mark.

4.2.2 PROGRAM FILES

A Program File contains code generated by the CHI-5 Linker, code generated manually via an editor, or the file may have received code as a Program Output File. The contents of the simulator's program memory may be loaded from, or stored to a Program File on a command from the user.

Each record in the file is in 3Z8 format (i. e. three fields of eight hexadecimal digits), and these records are grouped into blocks. The three numbers in the first record of a block represent 1) the block type (code block = type 1), 2) the base address in program memory (ROM or RAM) where the block is to be loaded, and 3) the number of instructions in the block. Each of the remaining records in the block contains a single instruction split into 3 numbers. The first number contains the high order 16 bits of the instruction (right justified), the second number contains the middle order 32 bits, and the third number contains the low order 32 bits.

4.2.3 STATE FILES

A State File contains values corresponding to simulator state variables. The file may have been generated via an editor, or it may have received values from the simulator as a State Output File. The state of the simulator may be loaded from, or stored to a State File on a command from the user.

This file is composed of a sequence of records. Each record contains one or more name-value pairs separated by at least one non-alphanumeric character. The first name-value pair in a record may be preceded by non-alphanumeric characters. The values of non-alphanumeric characters are ignored.

A name-value pair is composed of an alphabetic string (the name of a state variable in the simulation system) followed by a hexadecimal numeric string (the value to be assigned to the corresponding state variable). The two strings must be separated by at least one non-alphanumeric character. If a state variable name appears more than once in the file, then the last occurrence will determine the value assigned.

The file is terminated by an end-of-file mark.

4.3 LOGGING FILES

The only type of logging file is a Trace File. The values of state variables, whose trace flags are "true", are written to the Trace File whenever the state of the simulator changes (i.e. when an instruction is executed, or when the user modifies a state variable which has a true trace flag). So this file provides a selective record of the simulation which can be examined after the simulation is concluded.

The trace flags of state variables are set "true" or "false" by the user at any time during the simulation, but the values are written to the file only if a file name has been specified by command.

A Trace File is composed of a sequence of blocks, where each block corresponds to the state of the simulator between the execution of two instructions. Each block is composed of a sequence of records.

The first two records of a block contain the strings "PSA = xxxx" and "PHA = yyyy", where "xxxx" and "yyyy" are the values of PSA and PHA corresponding to the address of the next instruction to be executed. Each remaining record of a block contains name-value pairs separated by at least one non-alphanumeric character.

A name-value pair is composed of an alphabetic name followed by a hexadecimal string, the two strings being separated by "=". These pairs show the values of state variables immediately after the execution of an instruction.

5. COMMANDS

All commands and their arguments are entered on the System Input Device in response to the prompt "CHI5*". They are composed of one or more alphanumeric strings separated by any number of non-alphanumeric characters. File-names are an exception: they may contain non-alphanumeric characters, and are terminated by a space, comma, or end-of-line.

More than one fixed length command may be entered on the same line. If any syntax errors occur then the remaining commands on the line are aborted. Table 5.1 lists the commands in the order of their description.

```
C[ANCEL] B[REAK] address
C[ANCEL] C[OMMANDS]
C[ANCEL] D[ISPLAY] {A|State-Variable-Name}
C[ANCEL] T[RACE] {A|State-Variable-Name}
E[XIT]
G[OTO] address
L[OAD] A[NALOG] file-name
  NOL[OAD]
L[OAD] D file-name address [number]
L[OAD] P file-name
L[OAD] R file-name address [number]
L[OAD] S[TATE] file-name
L[OAD] X file-name address [number]
L[OAD] Y file-name address [number]
RES[ET]
R[UN]
SET state-variable-name value
SET B[REAK] address
SET D address value [value]...
SET DI[SPLAY] {A|state-variable-name}
SET P address valueA valueB valueC
              [valueA valueB valueC]...
SET R address value [value]...
SET TR[ACE] {A|state-variable-name}
SET X address value [value]...
SET Y address value [value]...
SHO[W] state-variable-name
SHO[W] A[DFIFO]
SHO[W] B[REAK]
SHO[W] D address [number]
SHO[W] DAF[IFO]
SHO[W] FI[LES]
SHO[W] P address [number]
SHO[W] R address [number]
SHO[W] S[TATE]
SHO[W] VA[RIABLES]
SHO[W] X address [number]
SHO[W] Y address [number]
```

Table 5.1-a LIST OF COMMANDS. ([] = Optional, {} = One element required, | = Exclusive or)

```
S[TEP] [number]
STO[RE] A[NALOG] file-name
  NOS[TORE]
STO[RE] D file-name address [number [address
 [number]]]...
STO[RE] P file-name address [number [address
 [number]]]...
STO[RE] R file-name address [number [address
 [number]]]...
STO[RE] S[TATE] file-name
STO[RE] X file-name address [number [address
 [number]]]...
STO[RE] Y file-name address [number [address
 [number]]]...
T[RACE] file-name
  NOT[RACE]
```

TABLE 5.1-b LIST OF COMMANDS. ([] = Optional, {} = One element required, | = Exclusive or)

4.1 C[ANCEL] B[REAK] address

Removes breakpoint address from breakpoint list.

4.2 C[ANCEL] C[OMMANDS]

1) Sets all state variable display and trace flags true.

2) Closes all files (except system input and system output files), setting file names to null.

3) Sets all state variables to undefined.

4.3 C[ANCEL] D[ISPLAY] {A|state-variable-name}
C[ANCEL] T[RACE] {A|state-variable-name}

"CANCEL DISPLAY A" sets all state variable display flags "false". If "A" is replaced by a state variable name, then the display flag for that state variable is set "false". Trace flags are affected in a similar manner.

4.4 E[XIT]

1) If the Trace File name is not null, then closes the Trace File.

2) Halts execution, returning control to the operating system.

4.5 G[OTO] address

1) Sets PSA (and BCNTR, if within the range of BCNTR) to the new address.

2) Fetches the new instruction.

4.6 L[OAD] A[NALOG] file-name
NOL[OAD]

"L[OAD] A[NALOG]":

1) Sets the new Analog Input File name.

2) Opens the file.

For succeeding analog clock ticks (once every 500 instructions) the next value will be read from the file and placed at the tail of the A/D FIFO.

"NOL[OAD]":

1) If the Analog Input File name is not null, then closes the file.

2) Sets the Analog Input File name null.

For succeeding analog clock ticks undefined values will be placed at the tail of the A/D FIFO.

4.7 L[OAD] D file-name address [number]

1) Sets the new Data Input File name.

2) Opens the file.

3) Reads "number" values from the file into D-memory, beginning at "address". If "number" is not given, then the default is the length of the file.

4) Closes the file.

4.8 L[OAD] P file-name

1) Sets the new Program Input File name.

2) Opens the file.

3) Reads the file into program memory.

4) Closes the file.

4.9 L[OAD] R file-name address [number]

1) Sets the new Table Input File name.

2) Opens the file.

3) Reads "number" values from the file into R-Memory, beginning at "address". If "number" is not given, then the default is the length of the file.

4) Closes the file.

4.10 L[OAD] S[TATE] file-name

1) Sets the new State Input File name.

2) Opens the file.

3) Reads the file, setting state variable values.

4) Closes the file.

4.11 L[OAD] X file-name address [number]
L[OAD] Y file-name address [number]

1) Sets the new X- or Y-Memory Input File name.

2) Opens the file.

3) Reads "number" values from the file into X- or Y-memory. If "number" is not given, then the default is the length of the file.

4) Closes the file.

4.12 RES[ET]

1) Initializes all state variables to an undefined value.

2) Sets PHA to 100 (HEX).

3) If Trace File name is not null, then writes values for state variables, whose trace flags are "true", to the Trace File.

4) Writes values for state variables, whose display flags are "true", to the System Output Device.

4.13 R[UN]

Places the simulator in RUN mode so that successive instructions are executed until a breakpoint is reached. After each instruction is executed the values of all state variables, whose trace flags are true, are written to the trace file (if it is open).

4.14 SET state-variable-name value

1) Sets the given state variable to the given value.

2) If the trace flag for the state variable is true and the trace file name is not null, then the value of all state variables whose trace flags are true are written to the trace file.

4.15 SET B[REAK] address

Places address in break point list.

4.16 SET D address value [value]...

Places value(s) in D-Memory beginning at "address".

4.17 SET DI[SPLAY] {A|state-variable-name}

"SET DISPLAY A" causes all display flags to be set "true". If "A" is replaced by a state-variable-name, then the display flag for that state-variable is set "true".

4.18 SET P address valueA valueB valueC
[valueA valueB valueC]...

Places value triples in program memory beginning at "address". Each value triple, valueA valueB valueC, corresponds to a single instruction. ValueA contains bits 79-64, valueB contains bits 63-32, and valueC contains bits 31-0 of the instruction.

4.19 SET R address value [value]...

Places value(s) in R-Memory beginning at "address".

4.20 SET TR[ACE] {A|state-variable-name}

"SET TRACE A" sets all trace flags "true". If "A" is replaced by a state-variable-name, then the trace flag for that state variable is set "true".

4.21 SET X address value [value]...
SET Y address value [value]...

Places value(s) in X- or Y-Memory beginning at "address".

4.22 SHO[W] state-variable-name

Displays the value of the given state variable.

4.23 SHO[W] A[DFIFO]

Displays the values in the A/D FIFO, in order, from head to tail.

4.24 SHO[W] B[REAK]

Displays all current entries in the breakpoint table.

4.25 SHO[W] D address [number]

Displays a segment of data memory of length "number" beginning at "address". The default for "number" is 1.

4.26 SHO[W] DAF[IFO]

Displays the values in the D/A FIFO, in order, from head to tail.

4.27 SHO[W] FI[LES]

Shows the current input and output file names for the State Variable and Logging files.

4.28 SHO[W] P address [number]

Displays a segment of program memory of length "number" beginning at "address". Each instruction is presented as a value triple. The default for "number" is 1.

4.29 SHO[W] R address [number]

Displays a segment of R-Memory of length "number" beginning at "address". The default for "number" is 1.

4.30 SHO[W] S[TATE]

Displays the values for all state variables whose display flags are "true".

4.31 SHO[W] VA[RIABLES]

Displays the current display and trace flags and value for each state variable.

4.32 SHO[W] X address [number]
SHO[W] Y address [number]

Displays a segment of X- or Y-Memory of length "number", beginning at "address". The default for "number" is 1.

4.33 S[TEP] [number]

Simulates the execution of "number" instructions. Default for "number" is 1.

4.34 STO[RE] A[NALOG] file-name
NOS[TORE]

"STO[RE] A[NALOG]":

1) Sets the new Analog Output File name.

2) Opens the file.

For succeeding analog clock ticks (once every 500 instructions) the head value in the D/A FIFO will be written to the Analog Output File.

"NOS[TORE]":

1) If the Analog Output File name is not null, then closes the file.

2) Sets the Analog Output File name null.

For succeeding analog clock ticks the head value in the D/A FIFO will not be written to a file.

4.35 STO[RE] D file-name address number [address number]...

Stores portions of D-memory in a file. Each portion begins at an "address" and is of length "number". The portions are placed in the file, one after the other, in the same order they are requested.

4.36 STO[RE] P file-name address number [address number]...

Stores portions of program memory in a file. Each portion begins at an "address" and is of length "number". The portions are placed in the file, one after the other, in the same order they are requested. Each instruction is written as a value triple.

4.37 STO[RE] R file-name address number [address number]...

Stores portions of R-Memory in a file. Each portion begins at an "address" and is of length "number". The portions are placed in the file, one after the other, in the same order they are requested.

4.38 STO[RE] S[TATE] file-name

Stores the values of all state variables in a file.

4.39 STO[RE] X file-name address number [address number]...
STO[RE] Y file-name address number [address number]...

Stores portions of X- or Y-memory in a file. Each portion begins at "address" and is of length "number". The portions are placed in the file, one after the other, in the same order they were requested.

4.40 T[RACE] file-name
NOT[RACE]

"TRACE":

1) If the Trace File name is not null, then closes the file.

2) Sets the Trace File name.

3) Opens the file.

For succeeding instruction executions, or other state changes, all state variables whose trace flags are "true" will have their values written to the file.

"NOT[RACE]:

1) If the Trace File name is not null, then closes the file.

2) Sets the Trace File name null.

State variables whose trace flags are "true" will not have their values written to the file.

APPENDIX A. INSTALLATION

The CHI-5 Simulator can be delivered on nine track tape recorded at 800, or 1600 BPI on a standard VAX-11/780. The FORTRAN-77 source files for the simulator are listed in Table A.1 along with a brief description.

All files should be copied to a mass storage device from which compilation and linkage may be performed. Those files which are not to be compiled must keep the same name when they are copied, because the names appear in INCLUDE statements in the remaining files.

Before compilation is performed, file CHI5PRMTR should be checked to ensure that:

1) The proper logical unit numbers have been assigned to the desired devices,

2) The record length of each device is appropriate (all record lengths have been set to 80 characters, but may be changed if desired), and

3) The status for the files on each device is appropriate (the status has been preset according to Table A.2, but may be changed if desired).

If the memory configuration of the CHI-5 Array Processor to be simulated is larger than that of the minimal machine, then file CHI5MEMRY must also be modified to reflect that configuration.

File Name	Description
CHI5BRKPNT	Breakpoint list for Outer System.
CHI5FMT	Format lists for Display and Trace transmissions.
CHI5IFLAG	Names and flags for Inner Simulator state variables.
CHI5IFLD	Operation fields for Inner Simulator.
*CHI5IINIT	Initializing routine for Inner Simulator.
CHI5IOPS	Instruction word for Inner Simulator.
*CHI5ISIM	Main routine for Inner Simulator.
CHI5IVALC	State variables for Inner Simulator.
CHI5IVALE	Named Inner Simulator state variables.
CHI5MEMRY	Parameters and declarations for simulator memory.
CHI5OFLAG	Names and flags for Outer Simulator state variables.
CHI5OFLD	Operation fields for Outer Simulator.
*CHI5OINIT	Initializing routine for Outer Simulator.
CHI5OOPS	Instruction word for Outer Simulator.
*CHI5OSIM	Main routine for Outer Simulator.
CHI5OVALC	State variables for Outer Simulator.
CHI5OVALE	Named Outer Simulator state variables.
CHI5PRMTR	Global parameters.
**CHI5SIM	User interface routines.
*CHI5SPPRT	Support routines.
*CHI5SYS	Communication routines.
CHI5TYPE	Instruction type variables for Outer Simulator.
CHI5USINT	State variables for user interface routines.

Table A.1 Files for CHI-5 Simulator and their contents.
(*=Files to be compiled; **=Main routine)

File Type	Logical Unit Parameter	Record Length Parameter	Status Parameter
Analog Input	AINDEV	ANRCLNTH	AINSTAT
Analog Output	AOUTDEV	ANRCLNTH	AOUTSTAT
D-Memory Input	DINDEV	DMRCLNTH	DINSTAT
D-Memory Output	DOUTDEV	DMRCLNTH	DOUTSTAT
Program Input	PINDEV	PMRCLNTH	PINSTAT
Program Output	POUDEV	PMRCLNTH	POUTSTAT
R-Memory Input	RINDEV	RMRCLNTH	RINSTAT
R-Memory Output	ROUTDEV	RMRCLNTH	ROUTSTAT
State Input	SINDEV	STRCLNTH	SINSTAT
State Output	SOUTDEV	STRCLNTH	SOUTSTAT
System Input	SYSIDEV	SYRCLNTH	SYSISTAT
System Output	SYSODEV	SYRCLNTH	SYSOSTAT
Trace Output	TRDEV	TRRCLNTH	TRSTAT
X-Memory Input	XINDEV	XMRCLNTH	XINSTAT
X-Memory Output	XOUTDEV	XMRCLNTH	XOUTSTAT
Y-Memory Input	YINDEV	YMRCLNTH	YINSTAT
Y-Memory Output	YOUTDEV	YMRCLNTH	YOUTSTAT

Table A.2 Parameter names, by file type, for logical units, record lengths, and file status.